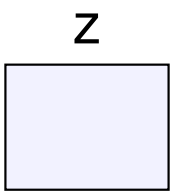
Arrays und Exceptions

Adrian Schüpbach
adrian_laurent.schuepbach@alumni.ethz.ch

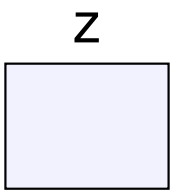
Arrays

- Variable deklarieren: `int[] z;`

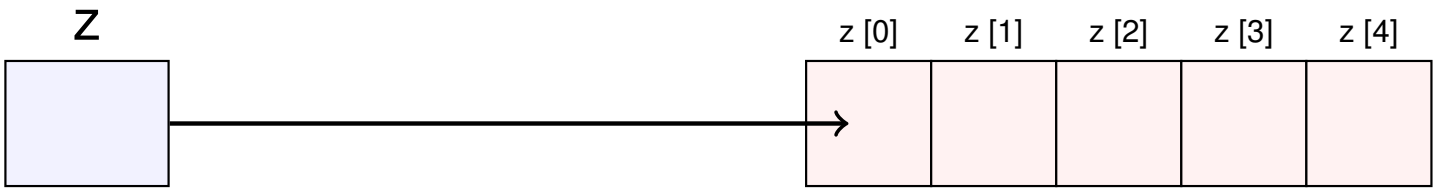


Arrays

- Variable deklarieren: `int[] z;`

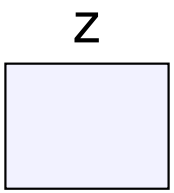


- Array erzeugen: `z = new int[5];`

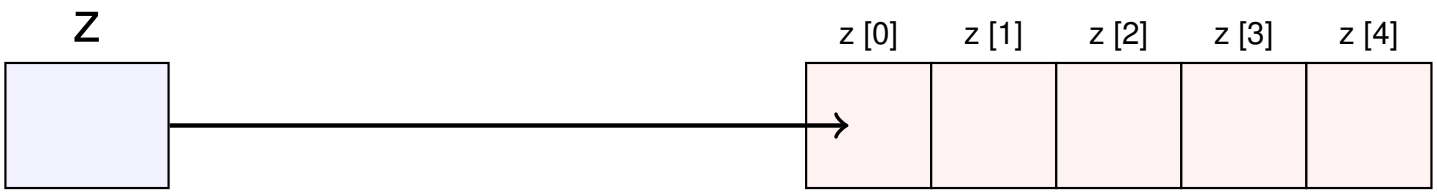


Arrays

- Variable deklarieren: `int[] z;`



- Array erzeugen: `z = new int[5];`



- `z` **zeigt** auf das Array...
  - ...aber erst, nachdem es erstellt wurde
  - ...ansonsten zeigt es nirgendwo hin: `null`
- Elemente werden indiziert
- Index beginnt bei **0** und endet bei **Arraygrösse - 1**

Arrays

Beispiel

- Aufgabe: Summe aller Elemente eines Arrays berechnen

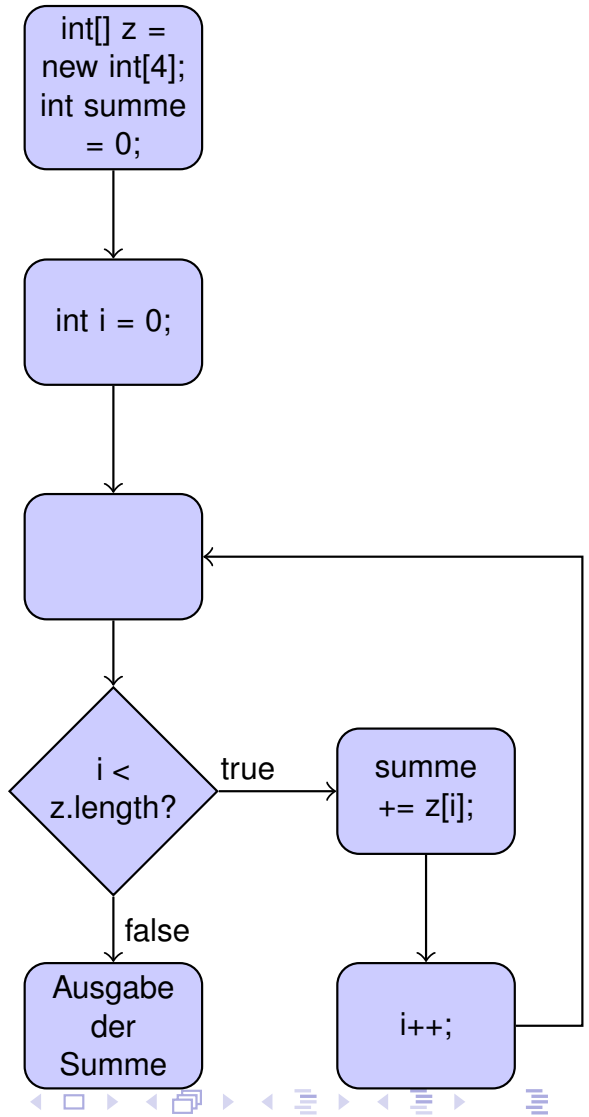
- Algorithmus
  1. Variable, die die Summe enthält, erstellen
  2. Variable mit 0 initialisieren
  3. Zählschleife (`for`-Schleife) erstellen
  4. Im Anweisungsblock
     - Zählvariable benutzen, um Array-Element zu lesen
     - Aktuelles Element zur Summenvariable addieren

Arrays

Beispiel

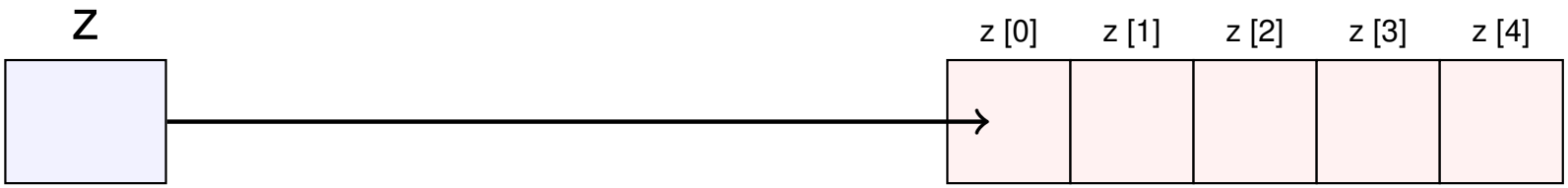
- Aufgabe: Summe aller Elemente eines Arrays berechnen

- Algorithmus
  1. Variable, die die Summe enthält, erstellen
  2. Variable mit 0 initialisieren
  3. Zählschleife (**`for`**-Schleife) erstellen
  4. Im Anweisungsblock
     - Zählvariable benutzen, um Array-Element zu lesen
     - Aktuelles Element zur Summenvariable addieren



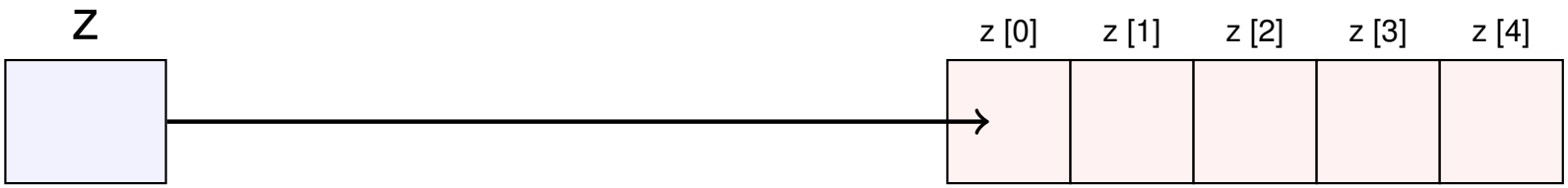
Zuweisungen von Arrays

```
int z[] = new int[5];
```



Zuweisungen von Arrays

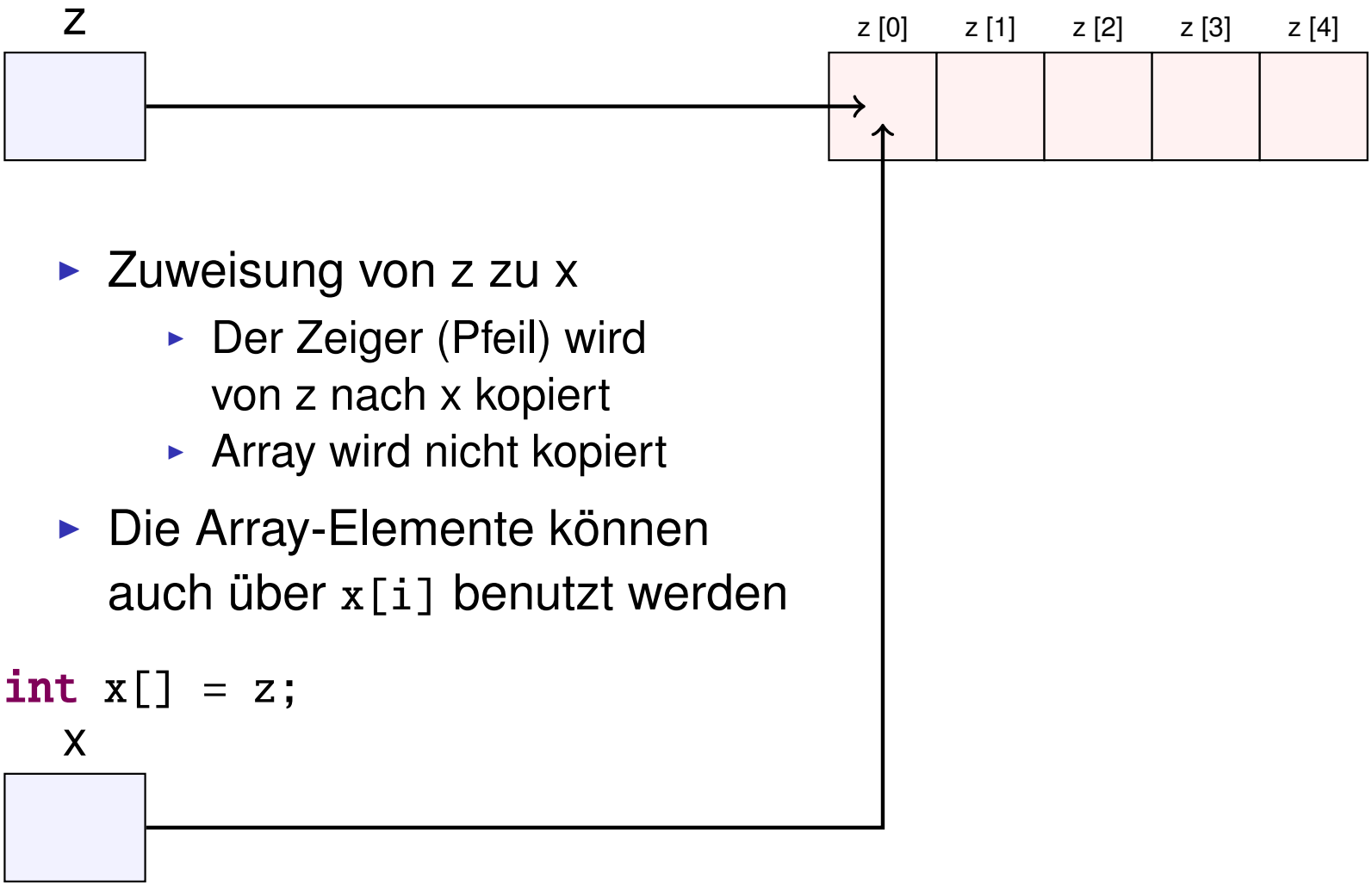
```
int z[] = new int[5];
```



- Zuweisung von z zu x
  - Der Zeiger (Pfeil) wird von z nach x kopiert
  - Array wird nicht kopiert
- Die Array-Elemente können auch über `x[i]` benutzt werden

Zuweisungen von Arrays

```
int z[] = new int[5];
```

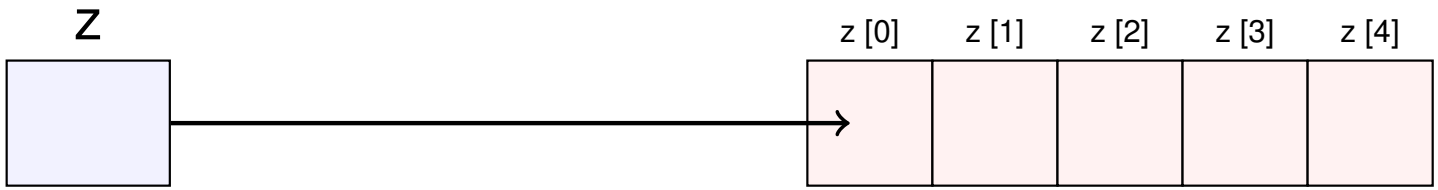


- Zuweisung von z zu x
  - Der Zeiger (Pfeil) wird von z nach x kopiert
  - Array wird nicht kopiert
- Die Array-Elemente können auch über `x[i]` benutzt werden

```
int x[] = z;
```

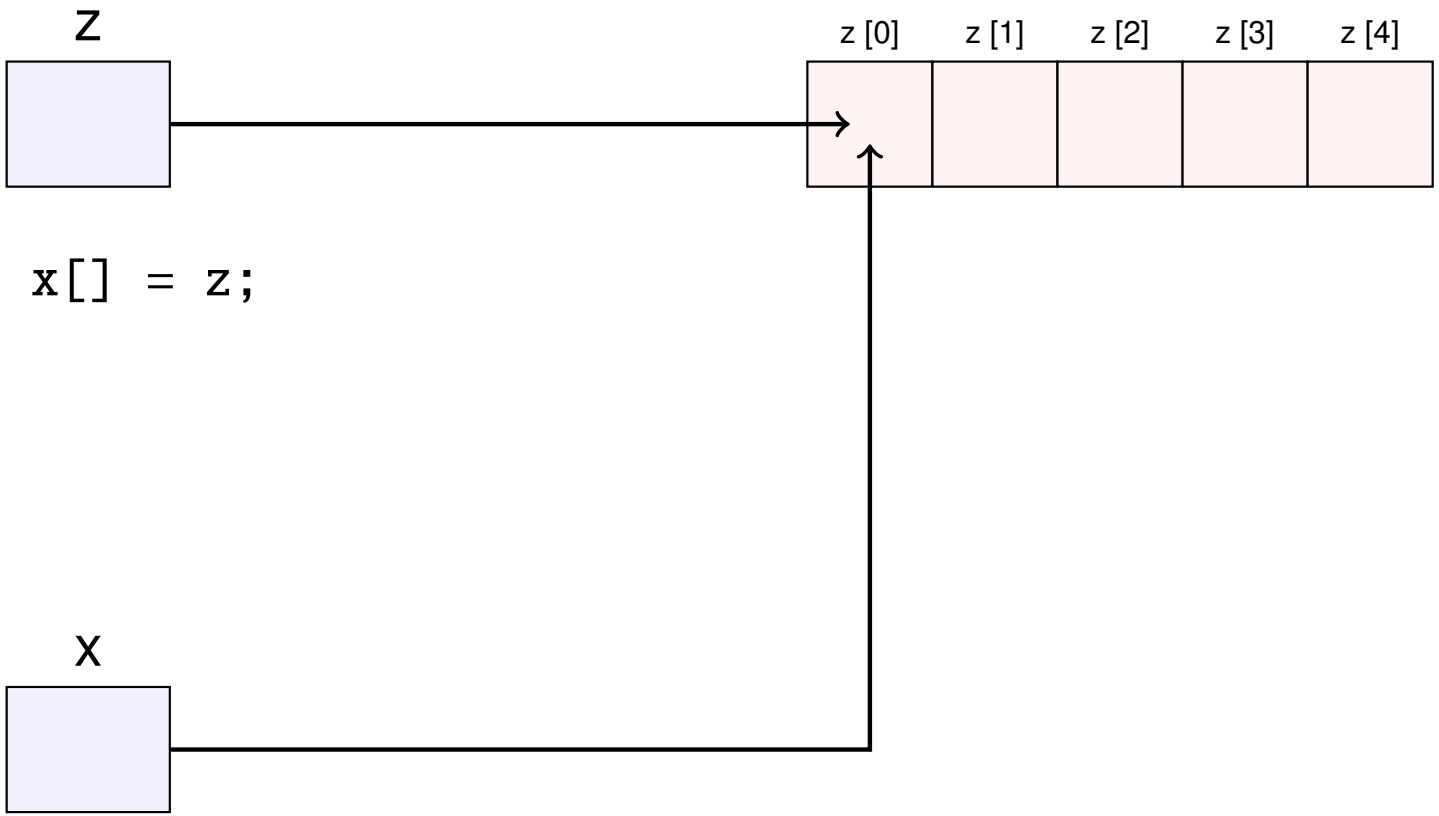
Zuweisungen von Arrays

```
int z[] = new int[5];
```



Zuweisungen von Arrays

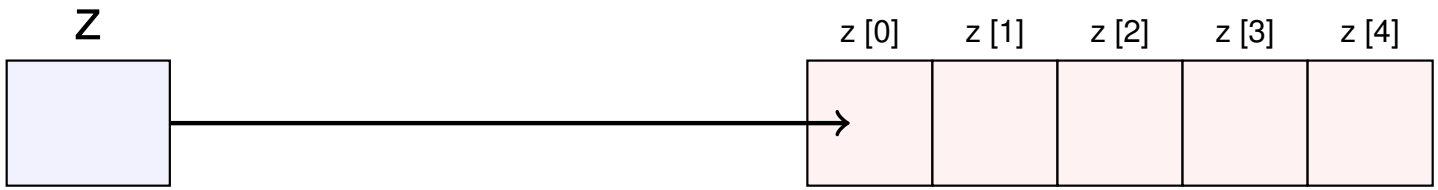
```
int z[] = new int[5];
```




```
int x[] = z;
```

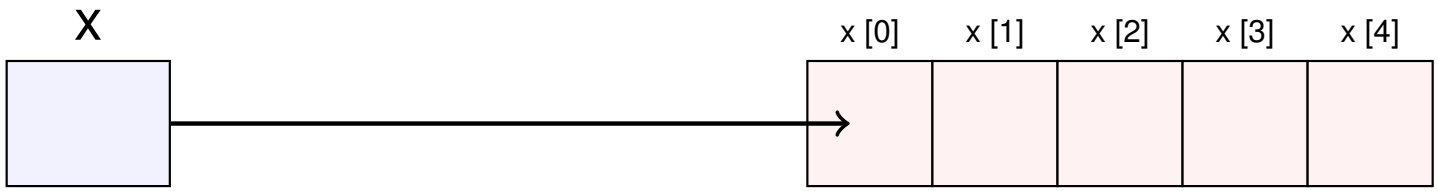
Zuweisungen von Arrays

```
int z[] = new int[5];
```



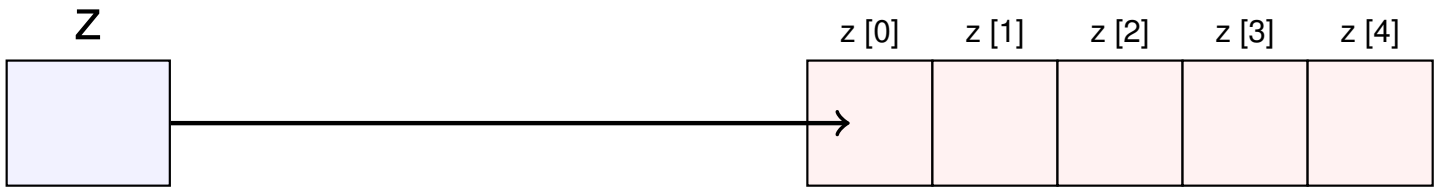

```
int x[] = z;
```

```
int x[] = new int[5];
```



Zuweisungen von Arrays

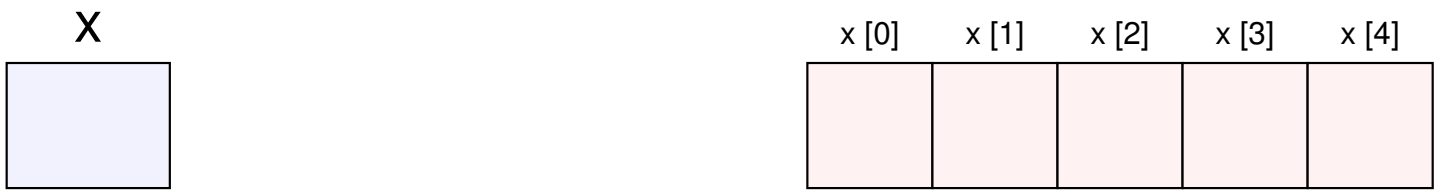
```
int z[] = new int[5];
```




```
int x[] = z;
```

```
int x[] = new int[5];
```

```
x = null;
```



Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6

- Gegeben:
  - Grosse Menge zufälliger oder gemessener Zahlen
  - Kleinste Zahl: 0
  - Grösste Zahl: 9
  - Zahlen in unsortiertem Array gegeben
- Gesucht
  - Wie oft kommt die Zahl 6 vor?

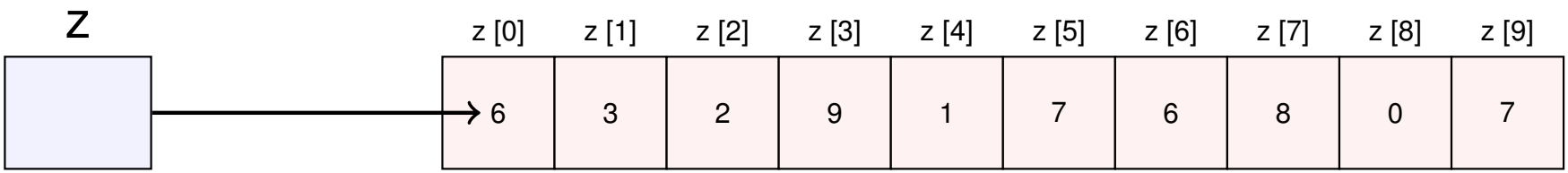
Beispiel: Wie oft kommt die Zahl N vor?

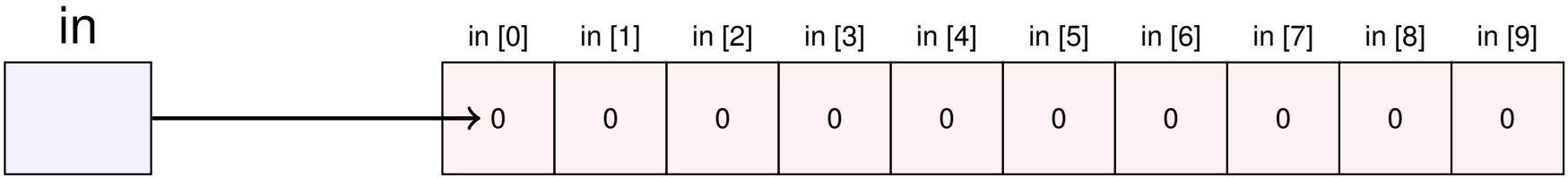
Z.Bsp. N = 6

- Algorithmus
  1. Erzeuge “Index-Array” mit höchstem Index 9
     - $\Rightarrow$ **10** Elemente!
  2. Schleife über alle gegebene Zahlen
     - Zahl als Index benutzen
     - “Index-Array” an diesem Index um eins erhöhen
  3. Im “Index-Array” bei Index 6 nachschauen, wie hoch der Zähler steht

Beispiel: Wie oft kommt die Zahl N vor?

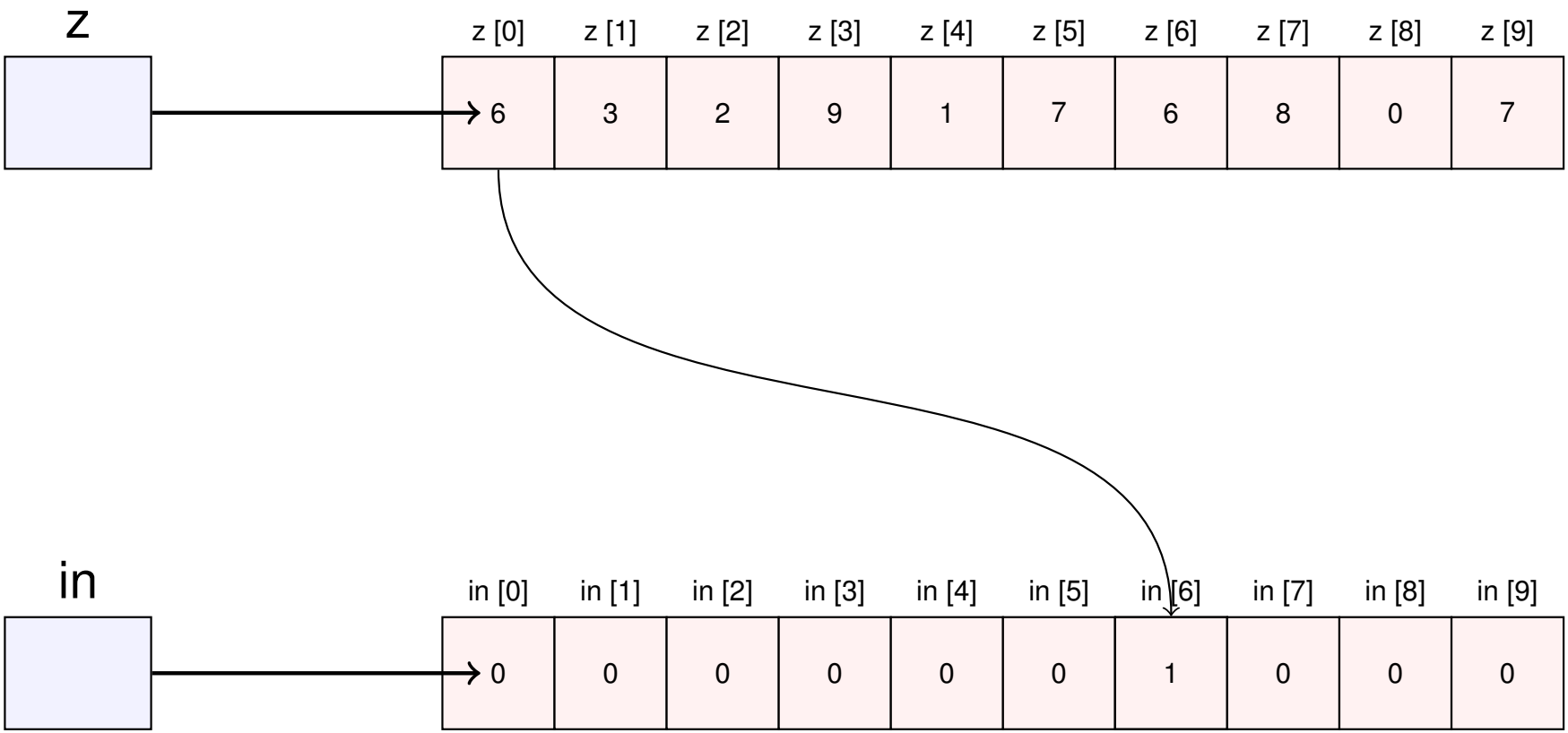
Z.Bsp. N = 6





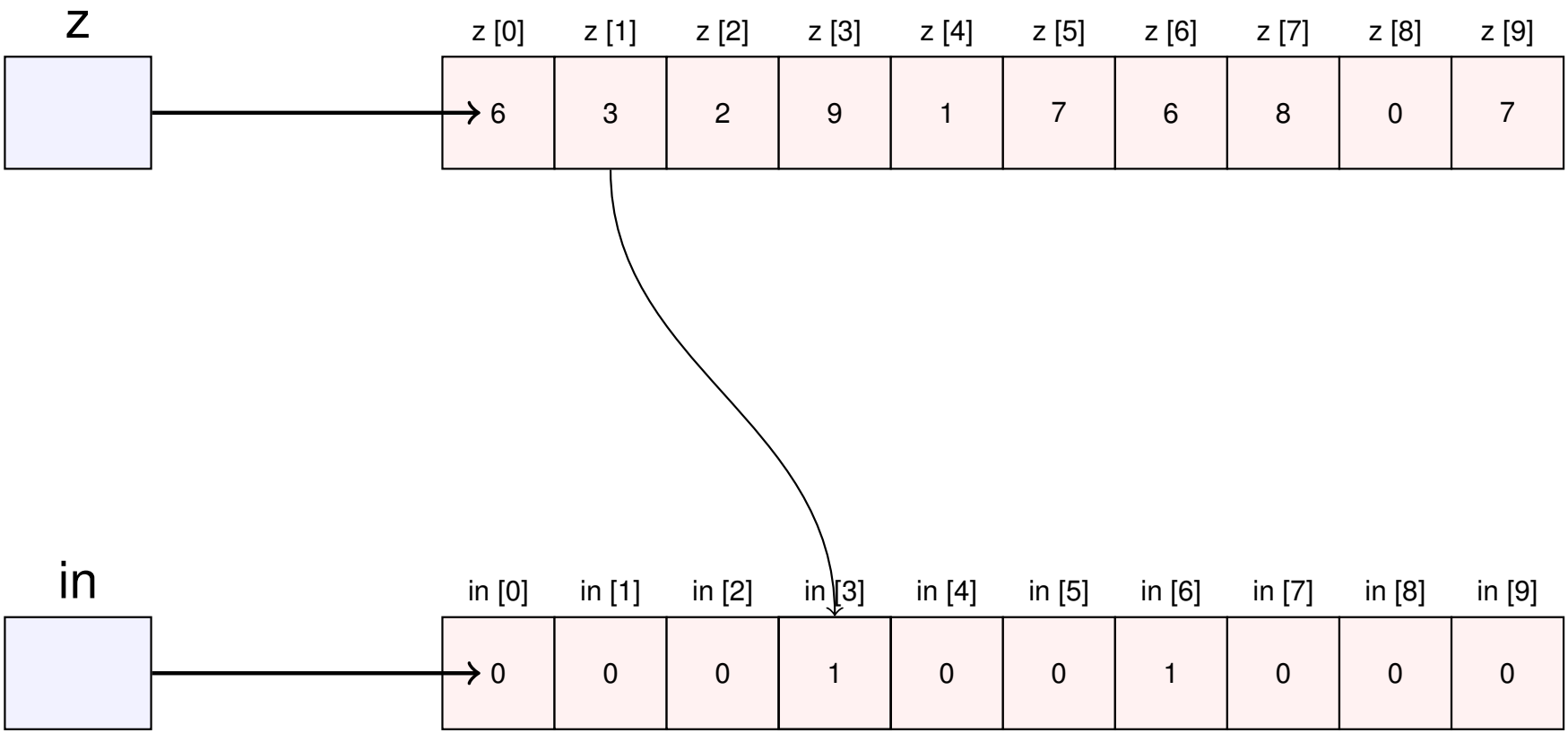
Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6



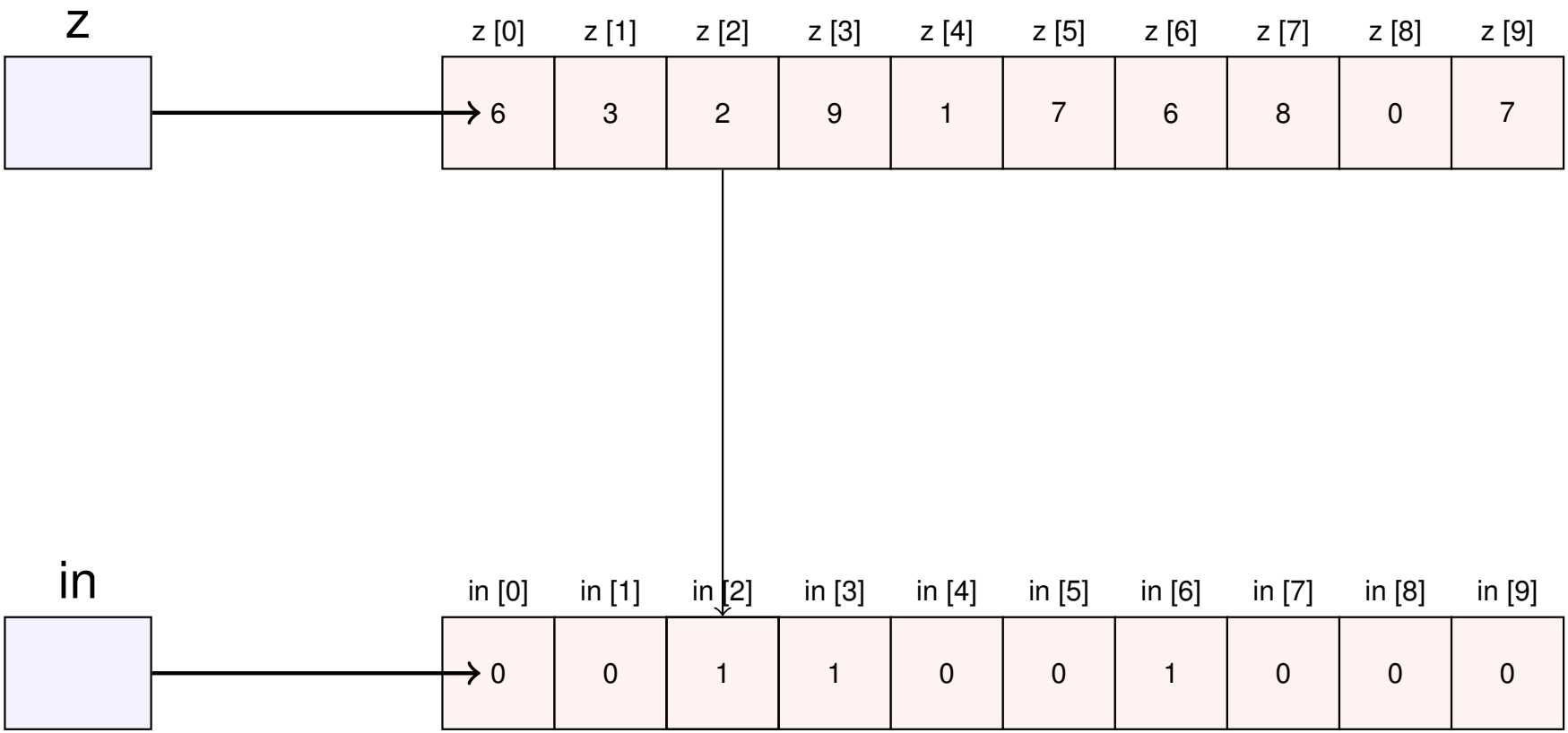
Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6



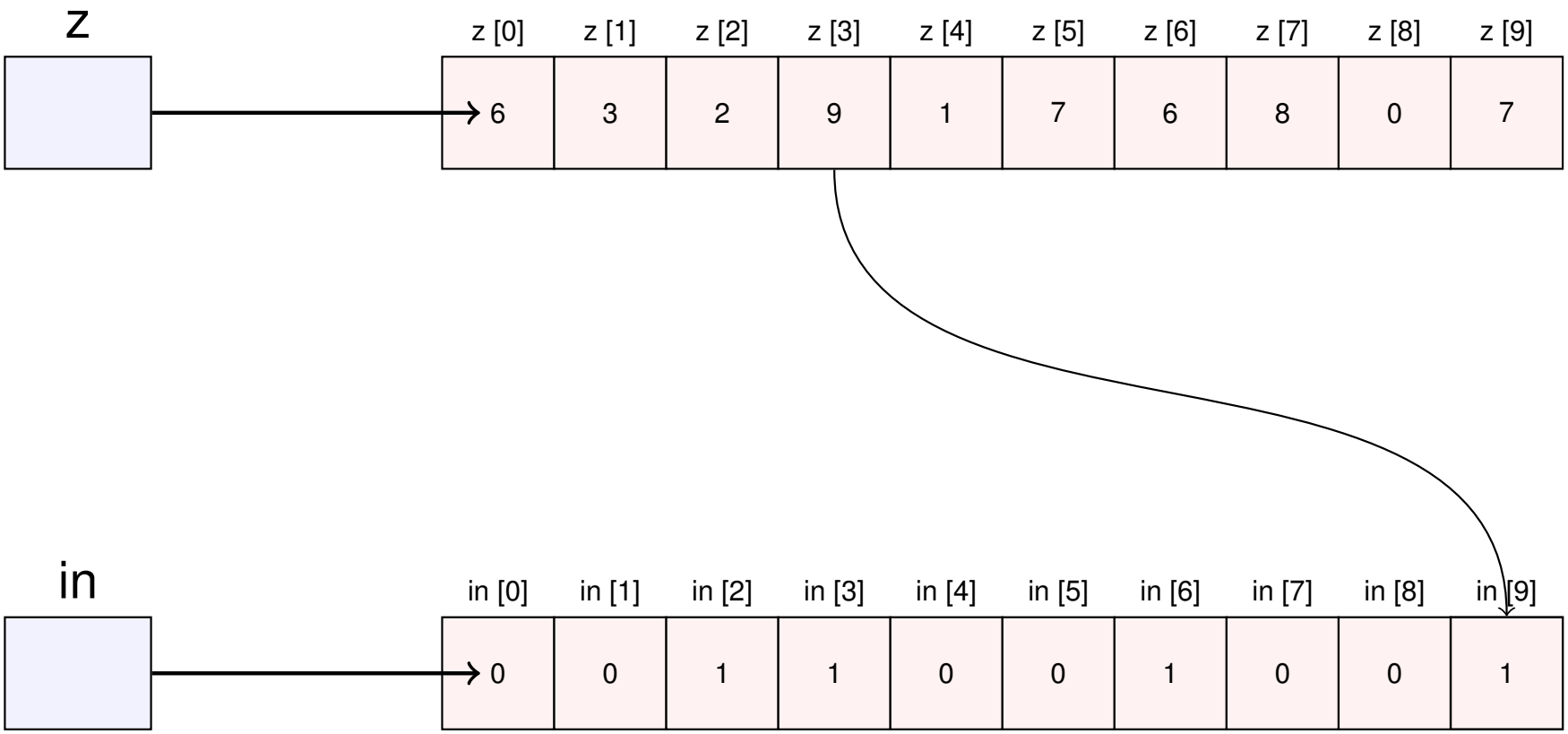
Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6



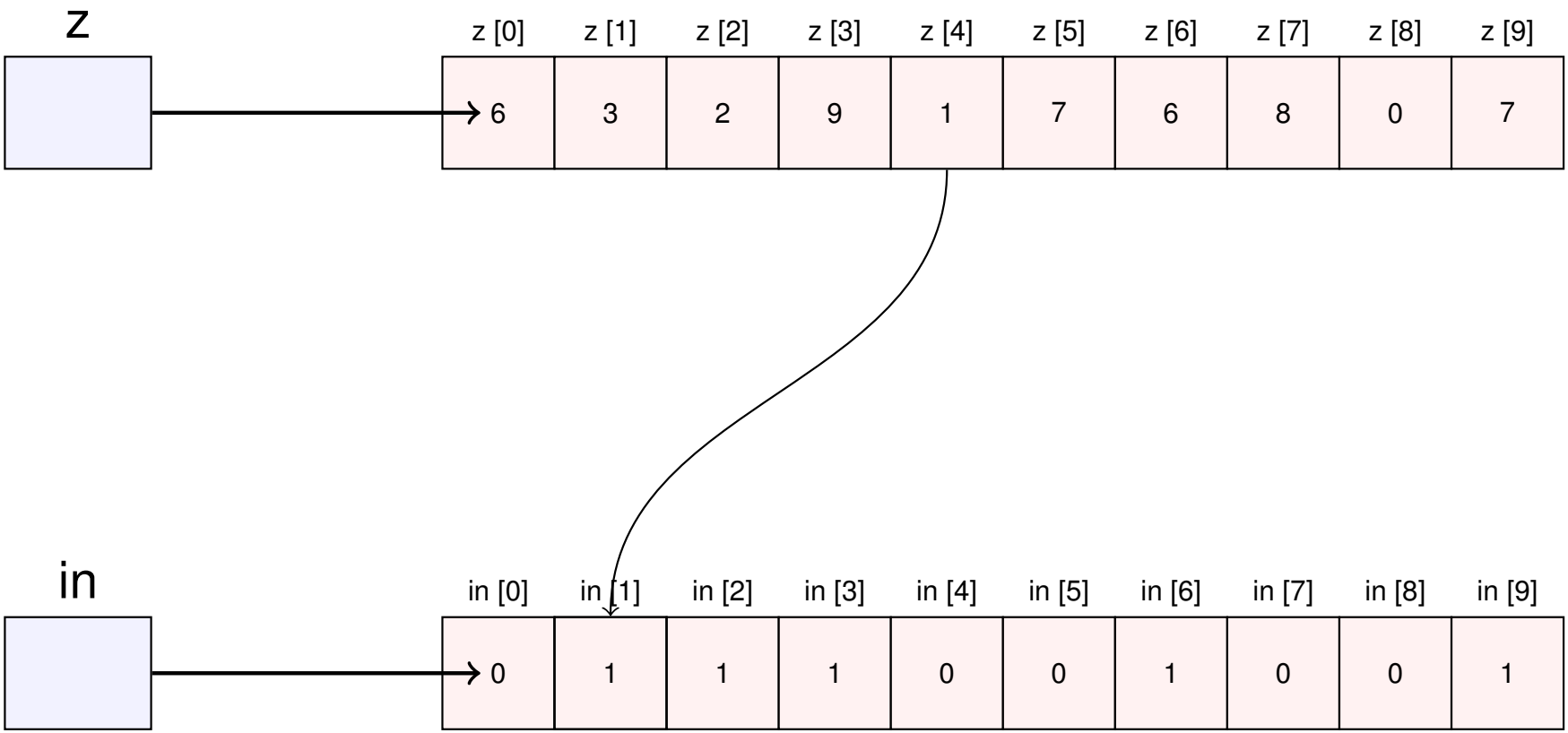
Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6



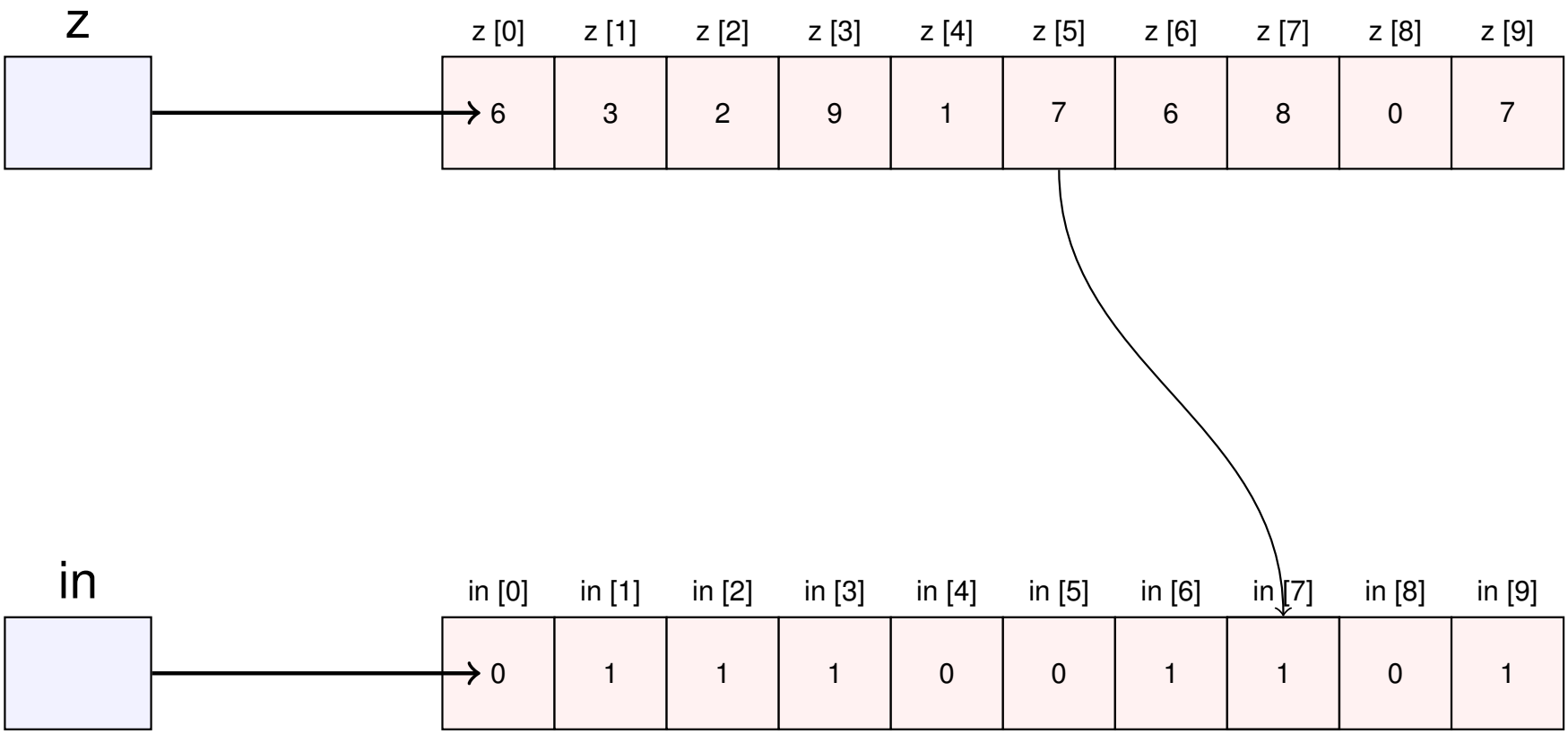
Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6



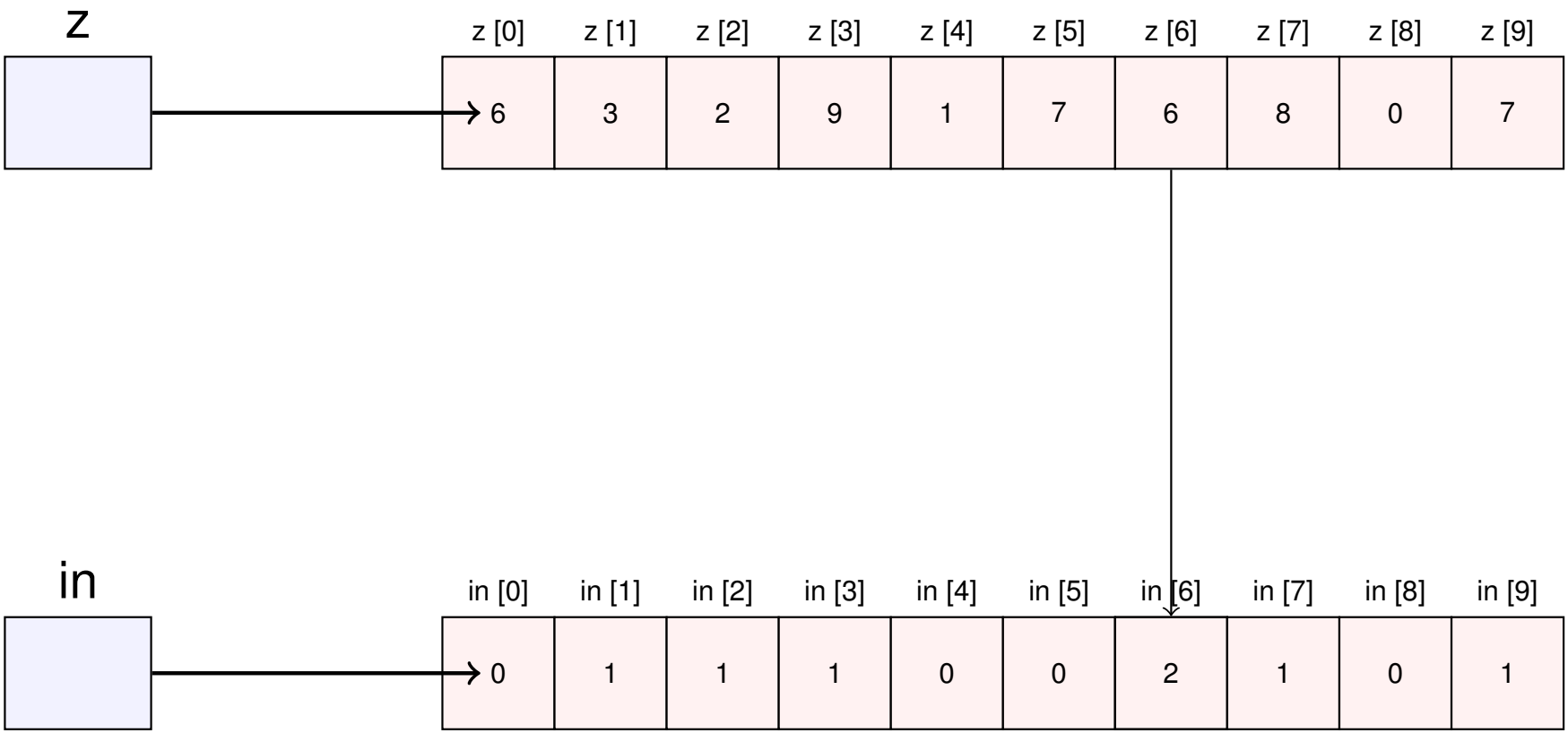
Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6



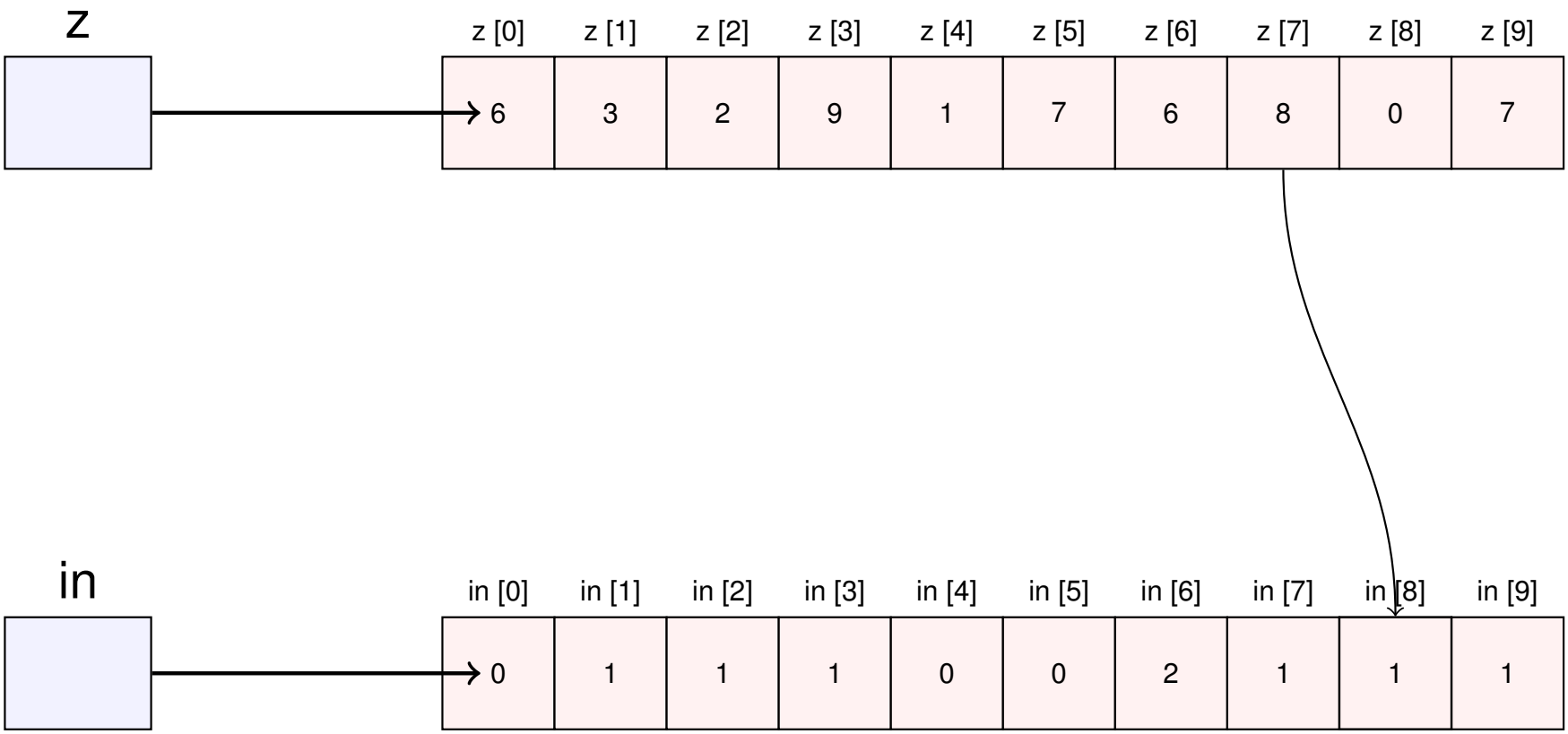
Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6



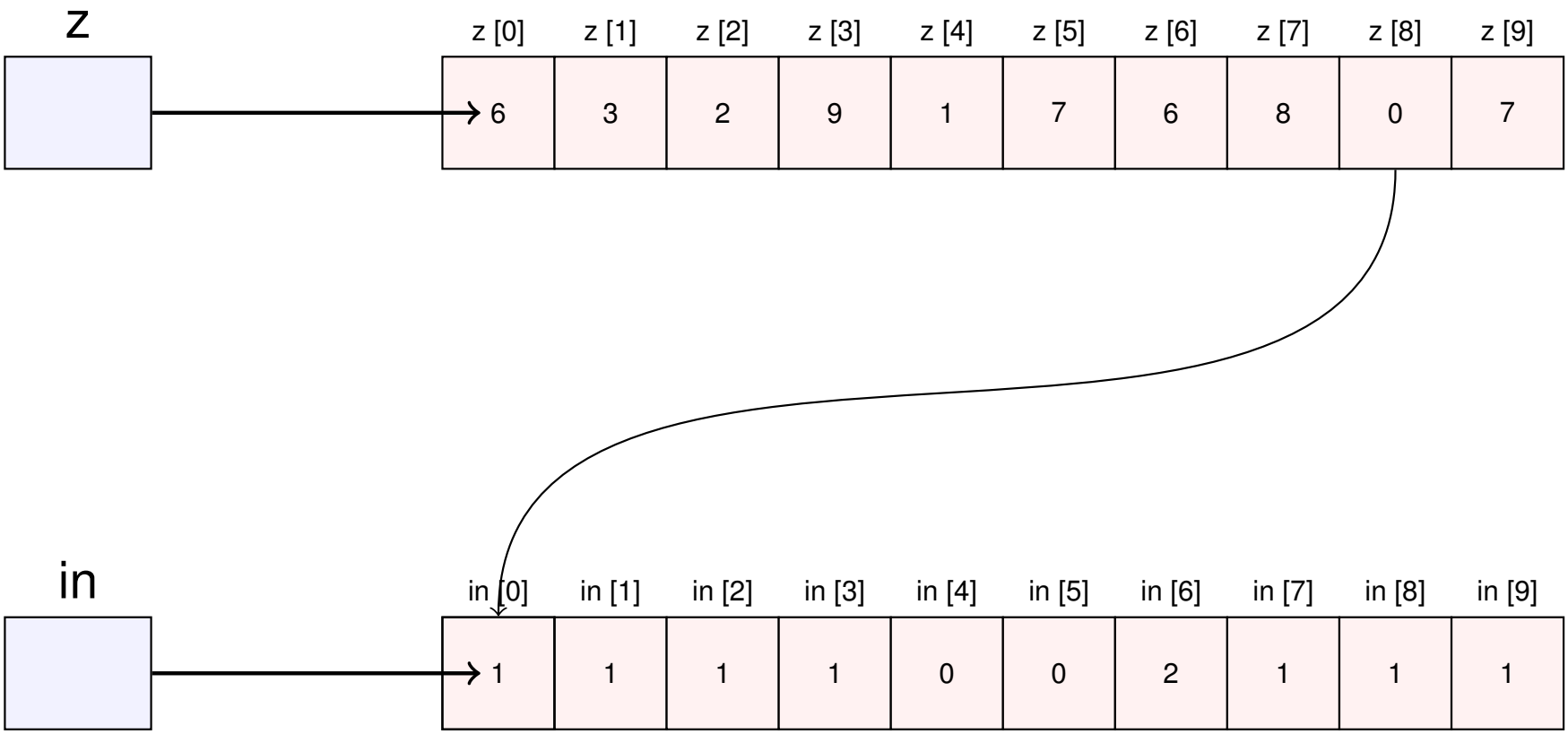
Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6



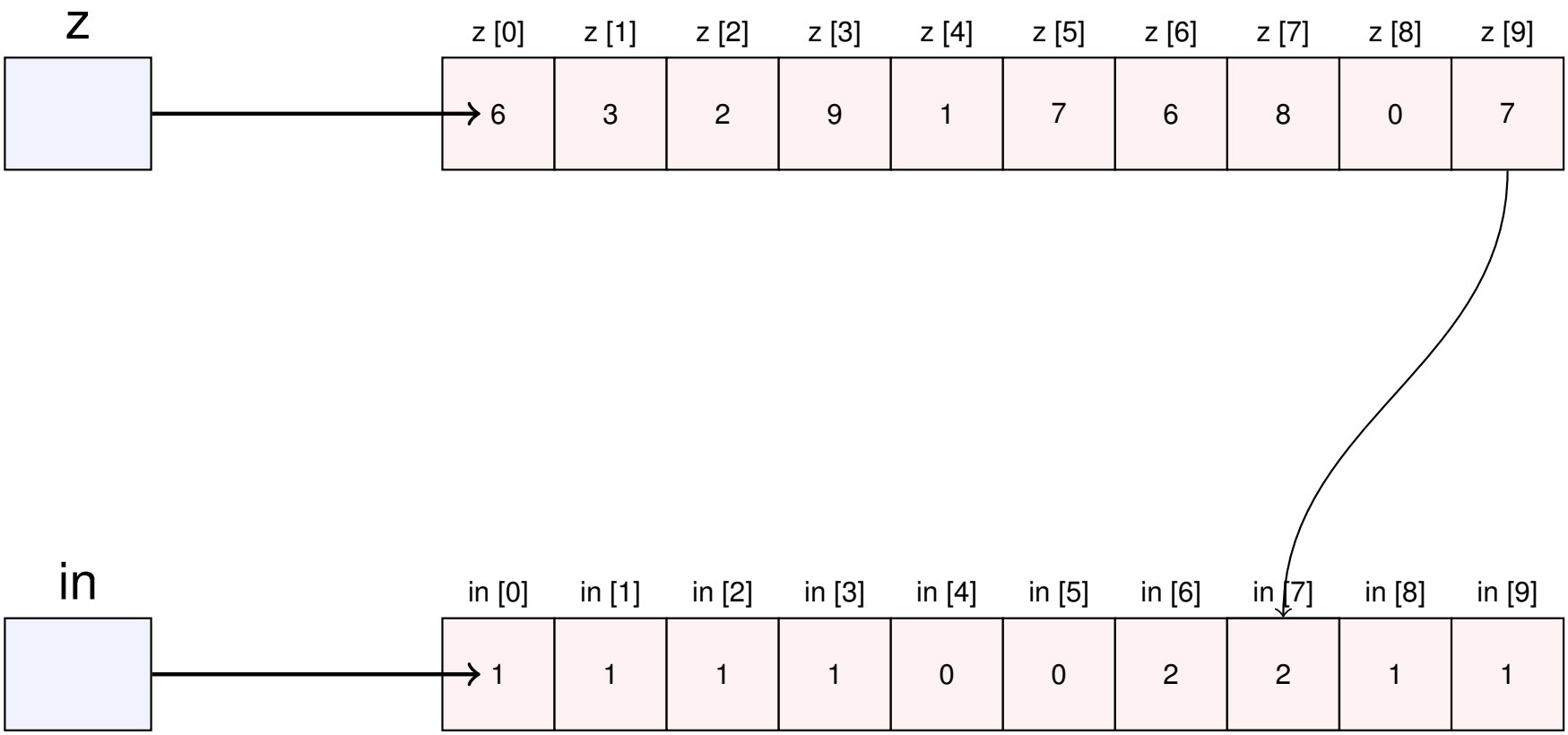
Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6



Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6



Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6

Zählen, wie oft jede Zahl vorkommt

```
public class Programm11 {
  public static void main(String[] args) {
    int gesuchte_zahl = 6;

    int[] zahlen = {6,3,2,9,1,7,6,8,0,7};
    int[] index = new int[10];

    // Schleife ueber alle gegebenen Zahlen
    for (int i = 0; i < zahlen.length; i++) {
      index[zahlen[i]]++;
    }
    System.out.println(gesuchte_zahl +" kommt " +
                       index[gesuchte_zahl] +
                       " mal vor.");
  }
```

Beispiel: Wie oft kommt die Zahl N vor?

Z.Bsp. N = 6

Kompakt

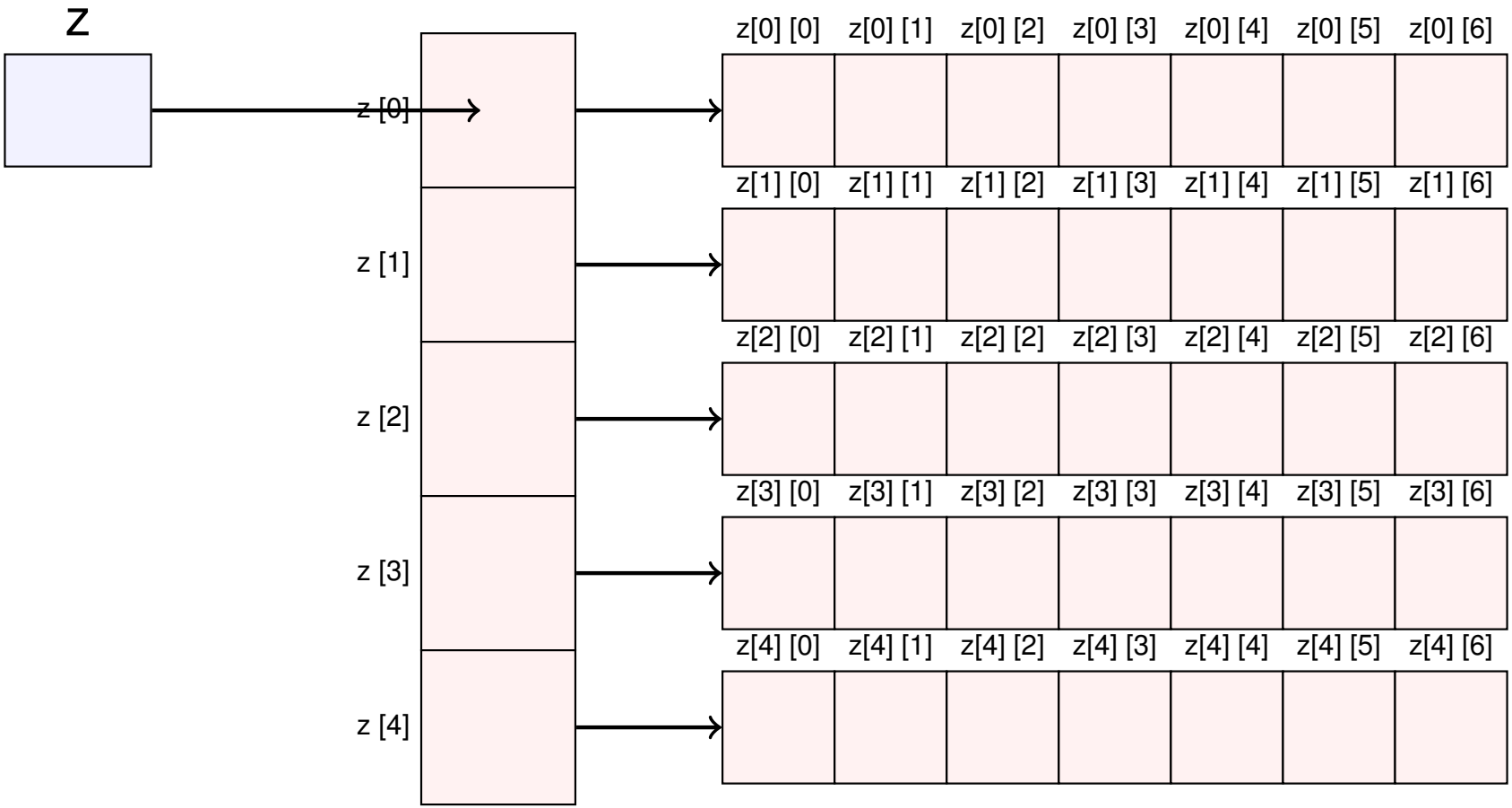
```
for (int i = 0; i < z... {
  //lese zahlen[i] und
  //benutze als Index in
  //index[], erhoehe um 1
  index[zahlen[i]]++;
}
```

Ausgeschrieben

```
for (int i = 0; i < z... {
  int zw = zahlen[i];
  int iw = index[zw];
  iw = iw + 1;
  index[zw] = iw;
}
```

Mehrdimensionale Arrays

```
int z[][] = new int[5][7];
```



Mehrdimensionale Arrays

- Zugriff auf ein Element (Zeile 3, Spalte 5):

```
int a = eineMatrix[3][5];
```

Mehrdimensionale Arrays

- Zugriff auf ein Element (Zeile 3, Spalte 5):

```
int a = eineMatrix[3][5];
```

- Zugriff auf eine ganze Zeile (Zeile 4):

```
int[] a = eineMatrix[4];
```

  - Zugriff ohne Spaltennummer ⇒ Ganze Zeile als Array benutzen
  - Rückgabe hier: `int[] zeile = eineMatrix[4];`
  - `zeile.length == 7`

Mehrdimensionale Arrays

Zugriff auf alle Elemente

- Alle Elemente der Reihe nach benutzen
  - Zwei verschachtelte `for`-Schleifen

Mehrdimensionale Arrays

Zugriff auf alle Elemente

- Alle Elemente der Reihe nach benutzen
  - Zwei verschachtelte `for`-Schleifen
  - Innere Schleife
    - Schleife über Spalten *einer* Zeile
    - Spaltennummer bei jeder Iteration erhöhen
    - Zeilennummer von äusseres Schleife gegeben
    - Zugriff auf Element

Mehrdimensionale Arrays

Zugriff auf alle Elemente

- Alle Elemente der Reihe nach benutzen
  - Zwei verschachtelte `for`-Schleifen
  - Innere Schleife
    - Schleife über Spalten *einer* Zeile
    - Spaltennummer bei jeder Iteration erhöhen
    - Zeilennummer von äusseres Schleife gegeben
    - Zugriff auf Element
  - Äussere Schleife
    - Schleife über alle Zeilen
    - Zeilennummer in jeder Iteration erhöhen
    - Innere `for`-Schleife in jeder Iteration aufrufen
- → Programm12

Ungültiger Index

- Ein Array beginnt immer bei Index 0
- Ein Array hat eine fixe Grösse

Ungültiger Index

- Ein Array beginnt immer bei Index 0
- Ein Array hat eine fixe Grösse

- Was passiert bei negativem oder zu grossem Index?

Ungültiger Index

- Ein Array beginnt immer bei Index 0
- Ein Array hat eine fixe Grösse

- Was passiert bei negativem oder zu grossem Index?

Exception in thread “main”
java.lang.ArrayIndexOutOfBoundsException: 10 at
test.Programm13.main(Programm13.java:6)

Mehrdimensionale Arrays: Welche Berge sind in der Nähe?

- Gegeben: Berge mit Koordinaten
- Gesucht: Berge, die innerhalb einer gegebenen Distanz zu einem gegebenen Punkt sind

Mehrdimensionale Arrays: Welche Berge sind in der Nähe?

- Gegeben: Berge mit Koordinaten
- Gesucht: Berge, die innerhalb einer gegebenen Distanz zu einem gegebenen Punkt sind

- Algorithmus
  - Schweiz in Koordinaten eingeteilt (CH1903)
  - Zweidimensionales Array, das Koordinaten in km enthält
  - Koordinaten der Berge lesen und Berg in richtiges "Quadrätchen" im Array speichern

Mehrdimensionale Arrays: Welche Berge sind in der Nähe?

- Array
  - Zeile $\Leftrightarrow$ x-Wert (km)
  - Spalte $\Leftrightarrow$ y-Wert (km)

Mehrdimensionale Arrays: Welche Berge sind in der Nähe?

- Array
  - Zeile $\Leftrightarrow$ x-Wert (km)
  - Spalte $\Leftrightarrow$ y-Wert (km)

- Problem: Was passiert, wenn zwei Berge sehr nahe sind?
  - Braucht zwei Einträge im gleichen “Quadrätchen”

Mehrdimensionale Arrays: Welche Berge sind in der Nähe?

- Array
  - Zeile $\Leftrightarrow$ x-Wert (km)
  - Spalte $\Leftrightarrow$ y-Wert (km)

- Problem: Was passiert, wenn zwei Berge sehr nahe sind?
  - Braucht zwei Einträge im gleichen “Quadrätchen”
- Lösung: Dreidimensionales Array
  - Zeile: x-Wert
  - Spalte: y-Wert
  - Dritte Dimension: Berg 0, 1, ..., n an der “gleichen” Stelle

Mehrdimensionale Arrays: Welche Berge sind in der Nähe?

- Abfragen
  - Benutzer gibt x- und y-Wert an
  - Könnte ein Array von mehreren Abfragen erstellen
- Abfragen verarbeiten
  - Programm: Direkter Zugriff auf erste **zwei** Dimensionen
  - $\rightarrow$ Resultat: Array mit Berg-IDs (dritte Dimension)
  - Schleife über dieses Array
  - Ausgabe der Berg-IDs

Mehrdimensionale Arrays: Welche Berge sind in der Nähe?

→ Demo: 1 Abfrage, mehrere Abfragen

Mehrdimensionale Arrays: Welche Berge sind grob innerhalb von 10km?

- Abfragen
  - Benutzer gibt Abfrage-Array an (x- und y-Werte)
  - Benutzer gibt Radius ein (z.Bsp. 10km)

Mehrdimensionale Arrays: Welche Berge sind grob innerhalb von 10km?

- Abfragen
  - Benutzer gibt Abfrage-Array an (x- und y-Werte)
  - Benutzer gibt Radius ein (z.Bsp. 10km)
- Wie macht man das?

Mehrdimensionale Arrays: Welche Berge sind grob innerhalb von 10km?

- Abfragen
  - Benutzer gibt Abfrage-Array an (x- und y-Werte)
  - Benutzer gibt Radius ein (z.Bsp. 10km)
- Wie macht man das?
- Lösung
  - Nicht fixen x-Wert, sondern Intervall von x-Werten
    - Hälfte des Radius vor und nach gegebenem x-Wert
  - Intervall von y-Werten

Mehrdimensionale Arrays: Welche Berge sind grob innerhalb von 10km?

- Abfragen
  - Benutzer gibt Abfrage-Array an (x- und y-Werte)
  - Benutzer gibt Radius ein (z.Bsp. 10km)
- Wie macht man das?
- Lösung
  - Nicht fixen x-Wert, sondern Intervall von x-Werten
    - Hälfte des Radius vor und nach gegebenem x-Wert
  - Intervall von y-Werten
  - Verschachtelte Schleifen, um x- und y-Intervalle zu durchlaufen

Mehrdimensionale Arrays: Welche Berge sind in der Nähe?

→ Demo

Design “Bergdatenstruktur”

- Vorteile

Design “Bergdatenstruktur”

- Vorteile
  - Schnelle Datenstruktur
  - Kann direkt an richtige Stelle im Array Werte auslesen
  - Einfach zu programmieren
- Nachteile

Design “Bergdatenstruktur”

- Vorteile
  - Schnelle Datenstruktur
  - Kann direkt an richtige Stelle im Array Werte auslesen
  - Einfach zu programmieren
- Nachteile
  - Zu grosser Speicherverbrauch
  - Westlichster und südlichster Teil des Arrays nie benutzt

Design “Bergprogramm”

- Radius 1000km sollte alle Berge enthalten
- → Demo: Versuch mit Radius 1000km

Design “Bergprogramm”

- Radius 1000km sollte alle Berge enthalten
- $\rightarrow$ Demo: Versuch mit Radius 1000km
- Exception in thread “main” java.lang.ArrayIndexOutOfBoundsException: -296 at berga.NachbarBerge10km.main(NachbarBerge10km.java:23)
- $\Rightarrow$ Programm sollte nur innerhalb des Koordinatensystems nach Bergen suchen

Design “Bergprogramm”

- Radius 1000km sollte alle Berge enthalten
- $\rightarrow$ Demo: Versuch mit Radius 1000km
- Exception in thread “main” java.lang.ArrayIndexOutOfBoundsException: -296 at berga.NachbarBerge10km.main(NachbarBerge10km.java:23)
- $\Rightarrow$ Programm sollte nur innerhalb des Koordinatensystems nach Bergen suchen

- Zwei Lösungsansätze:

Design “Bergprogramm”

- Radius 1000km sollte alle Berge enthalten
- $\rightarrow$ Demo: Versuch mit Radius 1000km
- Exception in thread “main” java.lang.ArrayIndexOutOfBoundsException: -296 at berga.NachbarBerge10km.main(NachbarBerge10km.java:23)
- $\Rightarrow$ Programm sollte nur innerhalb des Koordinatensystems nach Bergen suchen

- Zwei Lösungsansätze:
  - Zusätzliche UND-Bedingung (halbe Lösung)
  - min und max berechnen, dann Schleife